CSE211Digital Design

Akdeniz University

Week3: Introduction to Digital Design

Assoc.Prof.Dr. Taner Danışman
tdanisman@akdeniz.edu.tr

Course program

Week 01	09/16/2024	Introduction
Week 02	09/23/2024	Digital Systems and Binary Numbers I
Week 03	09/30/2024	Digital Systems and Binary Numbers II
Week 04	10/07/2024	Boolean Algebra and Logic Gates I
Week 05	10/14/2024	Boolean Algebra and Logic Gates II
Week 06	10/21/2024	Gate Level Minimization
Week 07	10/28/2024	Karnaugh Maps
Week 08	11/04/2024	Midterm
Week 09	11/11/2024	Karnaugh Maps
Week 10	11/18/2024	Combinational Logic
Week 11	11/25/2024	Combinational Logic
Week 12	12/02/2024	Timing, delays and hazards
Week 13	12/09/2024	Synchronous Sequential Logic
Week 14	12/16/2024	Synchronous Sequential Logic

Number Systems

- Decimal Numbers
  - What does 5,634 represent?
  - Expanding 5,634:

$$
\begin{aligned}
5 \times 10^3 &= 5{,}000 \\
+\ 6 \times 10^2 &= \quad 600 \\
+\ 3 \times 10^1 &= \quad\ \ 30 \\
+\ 4 \times 10^0 &= \quad\ \ \ \ 4 \quad \rightarrow 5{,}634
\end{aligned}
$$

  - What is "10" called in the above expansion?
    - The radix.
  - What is this type of number system called?
    - Decimal.
  - What are the digits for decimal numbers?
    - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
  - What are the digits for radix-r numbers?
    - 0, 1, ..., r-1.

Example: Convert 46.6875_{10} To Base 2

- **Convert 46 to Base 2**

  46/2 = 23 remainder = 0

  23/2 = 11 remainder = 1

  11/2 = 5 remainder = 1

  5/2 = 2 remainder = 1

  2/2 = 1 remainder = 0

  1/2 = 0 remainder = 1

  Read off in reverse order: 101110_2

- **Convert 0.6875 to Base 2:**

  0.6875 * 2 = 1.3750 int = 1

  0.3750 * 2 = 0.7500 int = 0

  0.7500 * 2 = 1.5000 int = 1

  0.5000 * 2 = 1.0000 int = 1

  0.0000

  Read off in forward order: 0.1011_2

- **Join together with the radix point: 101110.1011_2**

Converting Among Octal, Hexadecimal, Binary

- **Octal (Hexadecimal) to Binary:**
  - **Restate the octal (hexadecimal) as three (four) binary digits, starting at radix point and going both ways**
- **Binary to Octal (Hexadecimal):**
  - **Group the binary digits into three (four) bit groups starting at the radix point and going both ways, padding with zeros as needed in the fractional part**
  - **Convert each group of three (four) bits to an octal (hexadecimal) digit**
- **Example: Octal to Binary to Hexadecimal**

$$6\quad 3\quad 5\ .\ 1\quad 7\quad 7_{8}$$

$$= 110\,|\,011\,|\,101\ .\ 001\,|\,111\,|\,111_{2}$$

$$= 1\,|\,1001\,|\,1101\ .\ 0011\,|\,1111\,|\,1(000)_2 \text{ (regrouping)}$$

$$= 1\quad 9\quad D\ .\ 3\quad F\quad 8_{16} \text{ (converting)}$$

BCD

- The code is also known as 8-4-2-1 code.
- This is because 8, 4, 2, and 1 are the weights of the four bits of the BCD code
- Since four binary bits are used the maximum decimal equivalent that may be coded is 15_{10} (i.e., 1111_2).
- But the maximum decimal digit available is 9_{10}
- Hence the binary codes 1010, 1011, 1100, 1101, 1110, 1111, representing 10, 11, 12, 13, 14, and 15 in decimal are never being used in BCD code
- These six codes are called **forbidden codes**

BCD

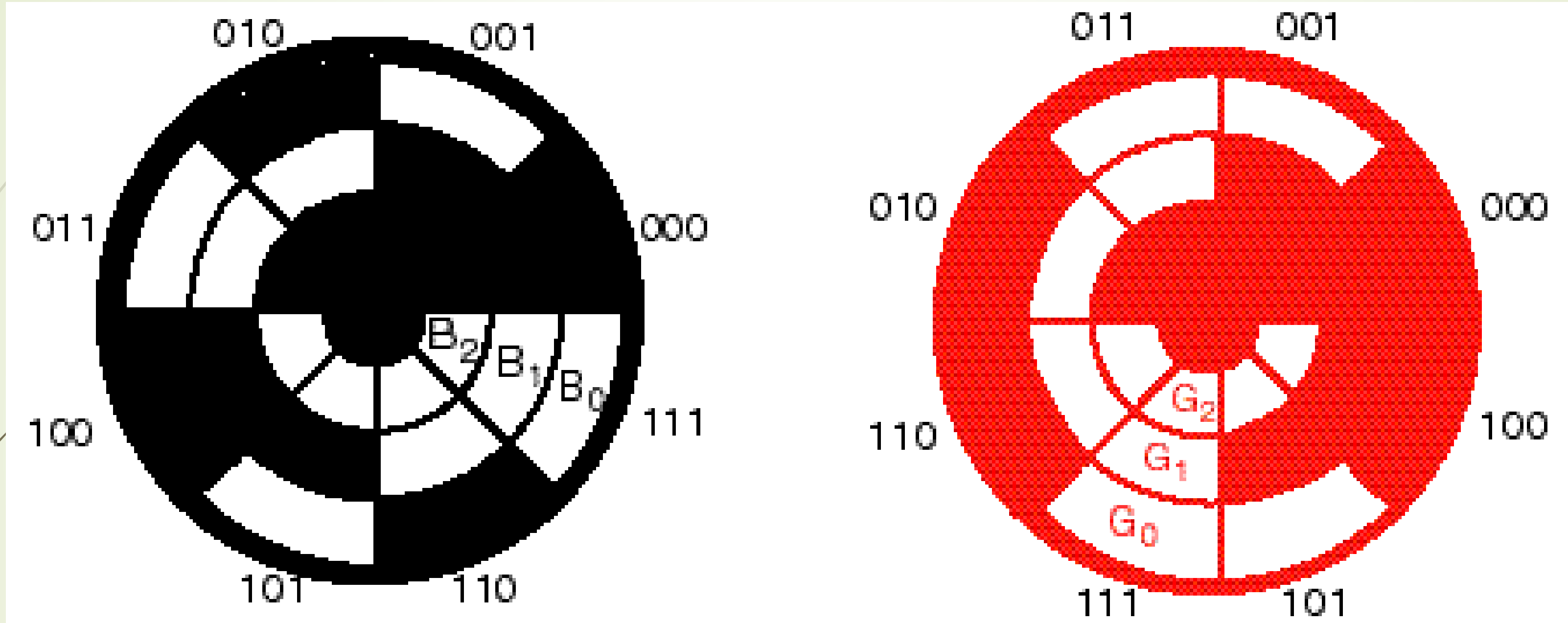
- **Example 2.1** Give the BCD equivalent for the decimal number 589.
- **Solution.**
  - The decimal number is 589 BCD code is 0101 1000 1001 Hence, $(589)_{10} = (010110001001)_{BCD}$

Gray Code

Decimal	8,4,2,1	Gray
0	0000	0000
1	0001	0100
2	0010	0101
3	0011	0111
4	0100	0110
5	0101	0010
6	0110	0011
7	0111	0001
8	1000	1001
9	1001	1000

- **What property does this Gray code have?**
  - Counting up or down changes only one bit at a time (including counting between 9 and 0)

Gray Code: Optical Shaft Encoder



(a) Binary Code for Positions 0 through 7

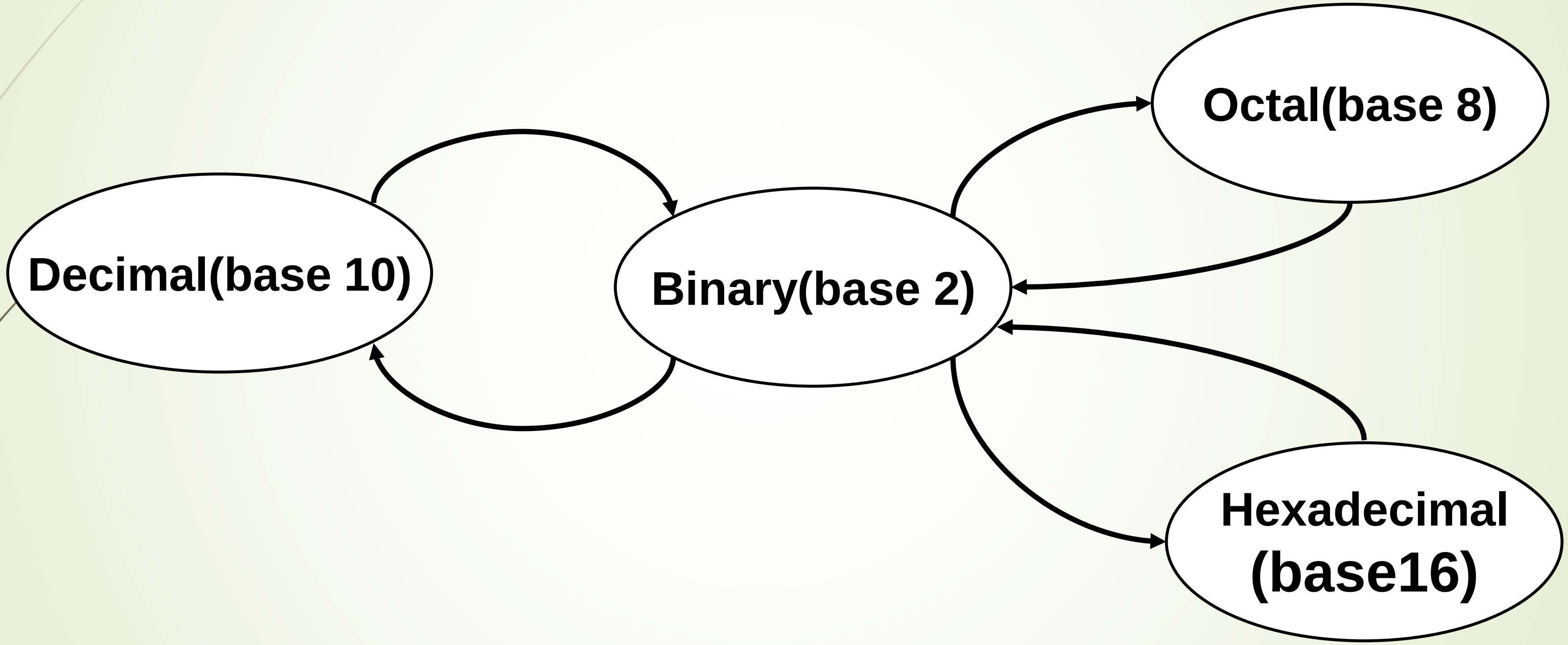
(b) Gray Code for Positions 0 through 7

- **Shaft encoder: Capture angular position (e.g., compass)**
- **For binary code, what values can be read if the shaft position is at boundary of “3” and “4” (011 and 100) ?**
- **For Gray code, what values can be read ?**

Warning: Conversion or Coding?

- **Do NOT mix up conversion of a decimal number to a binary number with coding a decimal number with a BINARY CODE.**
- **$13_{10} = 1101_2$ (This is conversion)**
- **13 $\Leftrightarrow$ 0001|0011 (This is coding)**

Conversion Between Number Bases



Conversion of a Binary Number into Gray Code

- the MSB of the Gray code is the same as the MSB of the binary number;
- the second bit next to the MSB of the Gray code equals the Ex-OR of the MSB and second bit of the binary number; it will be 0 if there are same binary bits or it will be 1 for different binary bits;
- the third bit for Gray code equals the exclusive-OR of the second and third bits of the binary number, and similarly all the next lower order bits follow the same mechanism

Conversion of a Binary Number into Gray Code

- **Example 2.2** *Convert* $(101011)_2$ *into Gray code.*

Solution.

Step 1. The MSB of the Gray code is the same as the MSB of the binary number.

1	0	1	0	1	1	Binary
↓						
1						Gray

Step 2. Perform the ex-OR between the MSB and the second bit of the binary. The result is 1, which is the second bit of the Gray code.

1 ⊕	0	1	0	1	1	Binary
	↓					
1	1					Gray

Step 3. Perform the ex-OR between the second and the third bits of the binary. The result is 1, which is the third bit of the Gray code.

1	0 ⊕	1	0	1	1	Binary
		↓				
1	1	1				Gray

Step 4. Perform the ex-OR between the third and the fourth bits of the binary. The result is 1, which is the fourth bit of the Gray code.

1	0	1 ⊕	0	1	1	Binary
			↓			
1	1	1	1			Gray

Step 5. Perform the ex-OR between the fourth and the fifth bits of the binary. The result is 1, which is the fifth bit of the Gray code.

1	0	1	0 ⊕	1	1	Binary
				↓		
1	1	1	1	1		Gray

Step 6. Perform the ex-OR between the fifth and the sixth bits of the binary. The result is 0, which is the last bit of the Gray code.

1	0	1	0	1 ⊕	1	Binary
					↓	
1	1	1	1	1	0	Gray

Conversion of a Binary Number into Gray Code

- Convert $(564)_{10}$ into Gray code.
- Decimal number 564 is equal to the Binary number $(1000110100)_2$
-

$1 \oplus$	$0 \oplus$	$0 \oplus$	$0 \oplus$	$1 \oplus$	$1 \oplus$	$0 \oplus$	$1 \oplus$	$0 \oplus$	0	Binary
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	
1	1	0	0	1	0	1	1	1	0	Gray

Some chapter terms to know

- DVD
- GUI
- BCD
- ASCII
- STX
- ETX
- CR
- LF
- HDL

Binary Arithmetic

- Single Bit Addition with Carry
- Multiple Bit Addition
- Single Bit Subtraction with Borrow
- Multiple Bit Subtraction
- Multiplication
- BCD Addition

Binary addition

- Same as decimal addition
  - 0+0=0
  - 0+1=1
  - 1+0=1
  - 1+1=0+C

0 + 0 0	0 + 1 1	1 + 0 1	1 + 1 10	1 1 + 1 11	1 +10 11	10 +101 111	1011 + 10 1101	101011 + 111 110010	1 1 1 1 101011 + 111 110010

Binary Addition

- Binary addition is very simple.
- This is best shown in an example of adding two binary numbers…

```
  111111      carries
   111101
+   10111
---------
  1010100
```

Binary Subtraction

- 0-0=0
- 1-0=1
- 1-1=0
- 0-1=0+B
- B(Borrow)

$\begin{array}{r} 0 \\ -\ 0 \\ \hline 0 \end{array}$	$\begin{array}{r} 1 \\ -\ 0 \\ \hline 1 \end{array}$	$\begin{array}{r} 1 \\ -\ 1 \\ \hline 0 \end{array}$	$\begin{array}{r} 10 \\ -\ 1 \\ \hline 01 \end{array}$	$\begin{array}{r} 101 \\ -\ 10 \\ \hline 011 \end{array}$

$$\begin{array}{r} 11001010 \\ -\ 10011011 \\ \hline 00101111 \end{array}$$

$$\begin{array}{r} 1001 \\ -\ 10 \\ \hline 0111 \end{array}$$

Binary Subtraction

- We can also perform subtraction (with borrows in place of carries).
- Let's subtract $(10111)_2$ from $(1001101)_2$…

$$
\begin{array}{cccccccccl}
 & & 1 & & & 10 & & & & \\
 & 0 & \cancel{10} & 10 & 0 & \cancel{0} & 10 & & \longleftarrow & \text{borrows} \\
 & & & & & & & & & \\
 & \cancel{1} & \cancel{0} & \cancel{0} & \cancel{1} & \cancel{1} & \cancel{0} & 1 & & \\
- & & & 1 & 0 & 1 & 1 & 1 & & \\
\hline
 & & 1 & 1 & 0 & 1 & 1 & 0 & &
\end{array}
$$

Binary Multiplication

- Binary multiplication is much the same as decimal multiplication, except that the multiplication operations are much simpler…

```
              1  0  1  1   1
X                1  0  1   0
---------------------------
              0  0  0  0   0
           1  0  1  1  1
        0  0  0  0  0
     1  0  1  1  1
---------------------------
     1  1  1  0  0  1  1   0
```

1's and 2's Complements

- Complements are used in digital computers to simplify the subtraction operation and for logical manipulation.
- Simplifying operations leads to simpler, less expensive circuits to implement the operations
- There are two types of complements for each **base-r** system:
  - the **diminished radix complement (r-1's complement)**
  - the **radix complement (r's complement)**

One's Complement Representation

- The one's complement of a binary number involves inverting all bits.
  - 1's comp of 00110011 is 11001100
  - 1's comp of 10101010 is 01010101
- Called **diminished radix complement** by Mano
- For an n bit number N the 1's complement is $\mathbf{(2^n-1) - N}$
  - e.g. For decimal numbers **r=10** and **r-1=9** So **9's** complement of **N** is $\mathbf{(10^n-1)-N}$
  - In this case, $\mathbf{10^n}$ represents a number that consists of a single 1 followed by ***n*** **0's**
  - $\mathbf{10^n - 1}$ is a number represented by ***n*** **9's**
  - if $n = 4$, we have $10^4 = 10{,}000$ and $10^4 - 1 = 9999$
- To find negative of 1's complement number take the 1's complement.

$0\underline{0001100}_2 = 12_{10}$ $\qquad$ $1\underline{1110011}_2 = -12_{10}$

Sign bit $\quad$ Magnitude $\qquad$ Sign bit $\quad$ Magnitude

Two's Complement Representation

- The two's complement of a binary number involves inverting all bits and adding 1.
  - 2's comp of 00110011 is 11001101
  - 2's comp of 10101010 is 01010110
- Called radix complement by Mano
- For an n bit number N the 2's complement is **$(2^n$-1) – N + 1**
  - the 10's complement of decimal **2389** is **7610 + 1 = 7611** and is obtained by adding 1 to the 9's complement value
  - The 2's complement of binary **101100** is **010011 + 1 = 010100** and is obtained by adding 1 to the 1's-complement value
- To find negative of 2's complement number take the 2's complement.

$00001100_2 = 12_{10}$

Sign bit Magnitude

$11110100_2 = -12_{10}$

Sign bit Magnitude

Two's Complement Shortcuts

- Algorithm 1 – Simply complement each bit and then add 1 to the result.

  Finding the 2's complement of $(01100101)_2$ and of its 2's complement…

  ```
  N  = 01100101        [N] =   10011011
        10011010               01100100
     +         1        +             1
     ------------        ------------
      10011011               01100101
  ```

- Algorithm 2 – Starting with the least significant bit, copy all of the bits up to and including the first 1 bit and then complementing the remaining bits.

  N = 0 1 1 0 0 1 0 1

  [N] = 1 0 0 1 1 0 1 1

1's Complement Addition

- Using 1's complement numbers, adding numbers is easy.
- For example, suppose we wish to add +12 and +1
- Let's compute $(12)_{10} + (1)_{10}$

$(12)_{10} = +(1100)_2 \qquad = 01100_2$ in 1's comp.

$(1)_{10} = +(0001)_2 \qquad = 00001_2$ in 1's comp.

Step 1: Add binary numbers
Step 2: Add carry to low-order bit

		0	1	1	0	0
+		0	0	0	0	1
	0	0	1	1	0	1
Add carry						0
Final Result		0	1	1	0	1

1's Complement Subtraction

- Using 1's complement numbers, subtracting numbers is also easy.
- For example, suppose we wish to subtract +1 from +12
- Let's compute $(12)_{10}$ - $(1)_{10}$.

$(12)_{10} = +(1100)_2 \quad = 01100_2$ in 1's comp.

$(-1)_{10} = -(0001)_2 \quad = 11110_2$ in 1's comp.

```
                0 1 1 0 0
-               0 0 0 0 1
            -------------

                0 1 1 0 0
+               1 1 1 1 0
Add carry   -------------
              1 0 1 0 1 0
                        1
Final       -------------
Result          0 1 0 1 1
```

Step 1: Take 1's complement of 2nd operand

Step 2: Add binary numbers

Step 3: Add carry to low order bit

2's Complement Addition

- Using 2's complement numbers, adding numbers is easy.
- For example, suppose we wish to add +12 and +1
- Let's compute $(12)_{10} + (1)_{10}$

$(12)_{10} = +(1100)_2 = 01100_2$ in 2's comp.

$(1)_{10} = +(0001)_2 = 00001_2$ in 2's comp.

		0 1 1 0 0
Add	+	0 0 0 0 1

Final Result		0 0 1 1 0 1

Ignore (the leading carry bit)

Step 1: Add binary numbers
Step 2: Ignore carry bit

2's Complement Subtraction

- Using 2's complement numbers, follow steps for subtraction
- For example, suppose we wish to subtract +1 from +12
- Let's compute $(12)_{10}$ - $(1)_{10}$

  $(12)_{10} = +(1100)_2 \quad = 01100_2$ in 2's comp.

  $(-1)_{10} = -(0001)_2 \quad = 11111_2$ in 2's comp.

$\boxed{0\ 1\ 1\ 0\ 0}$

\- 0 0 0 0 1

2's comp 0 1 1 0 0

Add + 1 1 1 1 1

Final Result 1 0 1 0 1 1

Ignore Carry

Step 1: Take 2's complement of 2nd operand
Step 2: Add binary numbers
Step 3: Ignore carry bit

2's Complement Subtraction: Example # 2

- Let's compute $(13)_{10} – (5)_{10}$.

  $(13)_{10} = +(1101)_2 = (01101)_2$

  $(-5)_{10} = -(0101)_2 = (11011)_2$

- Adding these two 5-bit codes…

```
                     0 1 1 0 1
carry          +     1 1 0 1 1
                   -------------
                 1   0 1 0 0 0
```

- Discarding the carry bit, the sign bit is seen to be zero, indicating a correct result. Indeed,

  $(01000)_2 = +(1000)_2 = +(8)_{10}$.

2's Complement Subtraction: Example #3

- Let's compute $(5)_{10} – (12)_{10}$.

  $(-12)_{10} = -(1100)_2 = (10100)_2$

  $(5)_{10} = +(0101)_2 = (00101)_2$

- Adding these two 5-bit codes…

$$
\begin{array}{rccccc}
 & 0 & 0 & 1 & 0 & 1 \\
+ & 1 & 0 & 1 & 0 & 0 \\
\hline
 & 1 & 1 & 0 & 0 & 1
\end{array}
$$

- Here, there is **no carry bit** and the sign bit is 1. This indicates a negative result, which is what we expect. $(11001)_2 = -(7)_{10}$.

2's Complement Subtraction: Example #4

Given the two binary numbers $X = 1010100$ and $Y = 1000011$, perform the subtraction **(a)** $X - Y$ and **(b)** $Y - X$ by using 2's complements.

(a)

$$
\begin{array}{rr}
X = & 1010100 \\
\text{2's complement of } Y = & +\ \underline{0111101} \\
\text{Sum} = & 10010001 \\
\text{Discard end carry } 2^7 = & -\ \underline{10000000} \\
\textit{Answer: } X - Y = & 0010001
\end{array}
$$

(b)

$$
\begin{array}{rr}
Y = & 1000011 \\
\text{2's complement of } X = & +\ 0101100 \\
\text{Sum} = & 1101111
\end{array}
$$

There is no end carry. Therefore, the answer is $Y - X = -(\text{2's complement of } 1101111) = -0010001$.

10's Complement Subtraction: Example #5

- Let's compute $(72532)_{10} - (3250)_{10}$.

  99999-3250

  96749+1=96750

Using 10's complement, subtract 72532 − 3250.

$$
\begin{array}{rl}
M = & 72532 \\
\text{10's complement of } N = & +\ \underline{96750} \\
\text{Sum} = & 169282 \\
\text{Discard end carry } 10^5 = & -\ \underline{100000} \\
\textit{Answer} = & 69282
\end{array}
$$

10's Complement Subtraction: Example #6

- Let's compute $(3250)_{10} - (72532)_{10}$.

  99999-72532

  27467+1=27468

Using 10's complement, subtract 3250 − 72532.

$$
\begin{aligned}
M &= \ 03250 \\
\text{10's complement of } N &= +\ \underline{27468} \\
\text{Sum} &= \ 30718
\end{aligned}
$$

There is no end carry. Therefore, the answer is −(10's complement of 30718) = −69282

BCD Addition

- When the binary sum is equal to or less than 1001 (without a carry), the corresponding BCD digit is correct
- However, when the binary sum is greater than or equal to 1010, the result is an invalid BCD digit
- The addition of 6 = $(0110)_2$ to the binary sum converts it to the correct digit and also produces a carry as required

$$
\begin{array}{rrrrrr}
4 & 0100 & 4 & 0100 & 8 & 1000 \\
\underline{+5} & \underline{+0101} & \underline{+8} & \underline{+1000} & \underline{+9} & \underline{1001} \\
9 & 1001 & 12 & 1100 & 17 & 10001 \\
 & & & \underline{+0110} & & \underline{+0110} \\
 & & & 10010 & & 10111
\end{array}
$$

BCD Addition

- If the binary sum is greater than or equal to 1010, we add 0110 to obtain the correct BCD sum and a carry

Consider the addition of 184 + 576 = 760 in BCD:

BCD	1	1		
	0001	1000	0100	184
	+0101	0111	0110	+576
Binary sum	0111	10000	1010	
Add 6		0110	0110	
BCD sum	0111	0110	0000	760

BCD Subtraction

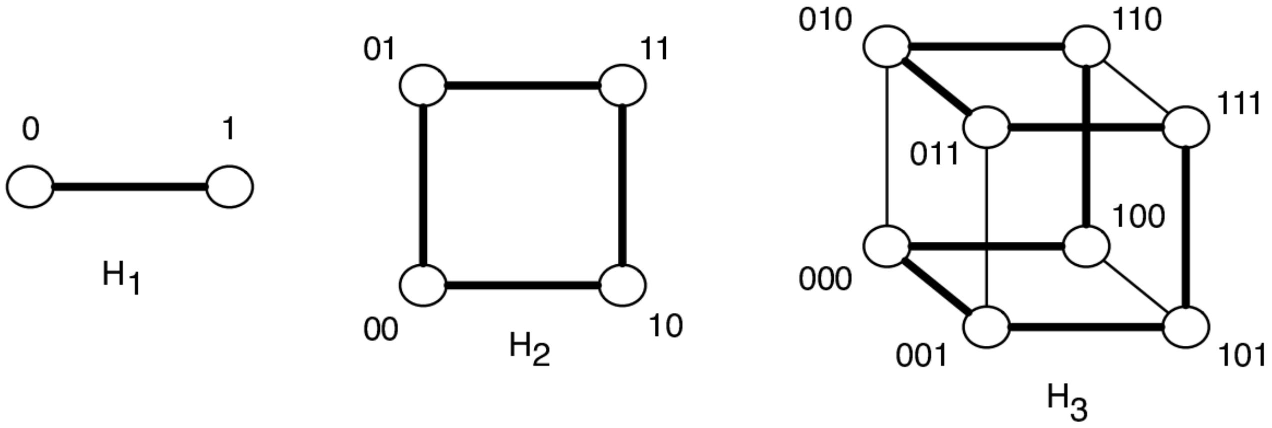
- We use 10's complement for the BCD subtraction. Positive numbers represented by **0000** while negative numbers represented by **1001**
- **357-432=?**
- 357 in BCD is 0000 0011 0101 0111
- -432 in **10**'s complement is 999 – 432 = 567, and 567 + 1 = 568
- Since this value is negatif we add trailing 1001 to 0101 0110 1000
- 0000 0011 0101 0111 + 1001 0101 0110 1000 = 1001 1000 1011 1111

0000	0011	0101	0111	+	1001	0101	0110	1000	=	1001	1000	1011	1111
0	3	5	7	+	9	5	6	8	=	9	8	11	15

- We add 6 to the nibble values greater than 1001.
- The we obtained **1001 1001 0010 0101** which is **-925**.
- In order to find the magnitude: **999 - 925 = 74**, and **74 + 1 = 75**
- Then final result is **357 - 432 = -75**

Questions

- What coding type might be used in the following figures?



- How can we find 16th gray code value?

Gray Code

Gray Code	Decimal Equivalent
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

Excess -3 Codes

- Simple add decimal 3 to the BCD value
- It is the only **unweighted code** which is **self complementing**
- **What are invalid digits?**

Decimal	BCD	XS-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Excess 3 Codes Adition

- 0011 0101 0110 + 0101 0111 1001=?
- Steps
  - Convert to BCD
  - Add 3 to each individual digit to find the Excess-3 code
  - Perform standard binary addition
  - **Add** 0011 to the groups that creates a carry and **subtract** 0011 from groups that does not create carry
  - Result is in EX-3 code so substract additional 3 from the result to find BCD equivalent

Excess-3 Subtraction

- **Step 1** Like the previous method both the numbers have to be converted into excess 3 code
- **Step 2** Following the basic methods of binary subtraction, subtraction is done.
- **Step 3** **Subtract** ‘0011’ from each BCD four-bit group in the answer if the subtraction operation of the relevant four-bit groups required a borrow from the next higher adjacent four-bit group.
- **Step 4** **Add** ‘0011’ to the remaining four-bit groups, if any, in the result.
- **Step 5** Finally we get the desired **result in excess 3 code**

Excess-3 Subtraction Example

- Let us take the numbers 0001 1000 0101 and 0000 0000 1000
- Excess 3 equivalent of those numbers are
  - 0100 1011 1000 and 0011 0011 1011

```
  0100 1011 1000
– 0011 0011 1011
----------------
  0001 0111 1101
```

- The least significant column which needed a borrow and the other two columns did not need borrow
  - Therefore, **subtract** 0011 from the result of this column and **add** 0011 to the other two columns
  - We get 0100 1010 1010
- This is the result expressed in excess 3 codes.
- The binary result is 0001 0111 0111